Predicción Salario Mayor a $50,000 USD al Año

Carlos Velasco, ITC A01708634, Tecnológico de Monterrey Campus Querétaro

Abstracto.- Este documento presenta la implementación y explicación sobre la determinación de salarios anuales mayores a $50,000 USD en Estados Unidos basado en datos del consenso.

I. Introducción

El ingreso anual es un indicador clave del bienestar económico de los individuos y las familias, especialmente en países como los Estados Unidos. Ganar $50,000 al año es un punto de referencia significativo que a menudo se utiliza para diferenciar entre diferentes niveles de estabilidad financiera.

Para muchas personas, ganar menos de $50,000 USD al año puede significar dificultades para mantenerse al día con los costos de vida, especialmente en áreas urbanas con altos gastos de vivienda. Por otro lado, superar este umbral puede representar un mayor acceso a oportunidades y una mejor calidad de vida.

El presente análisis tiene como objetivo explorar los factores demográficos y socioeconómicos que influyen en la probabilidad de que una persona gane más de $50,000 USD al año. Utilizando datos censales, se busca construir un modelo predictivo que permita entender mejor las características que diferencian a aquellos que superan este umbral de ingresos de aquellos que no lo hacen.

II. Proceso de Análisis

El análisis de datos comienza con la identificación y selección de las variables más relevantes para alcanzar el objetivo: predecir si el ingreso anual de una persona excede los $50,000 USD. Este paso es fundamental, ya que determina las características que influyen en la capacidad predictiva del modelo.

Paso 1: Definición de la Información Necesaria

El primer paso fue identificar qué información del conjunto de datos censales es esencial para la predicción. Se seleccionaron características clave que reflejan aspectos demográficos, laborales, y sociales, tales como la edad, clase de trabajo, nivel educativo, estado civil, ocupación, relación familiar, raza, sexo, horas trabajadas por semana, y el país de origen. Estas variables son determinantes en la estructura económica de los individuos y consideré que pueden influir significativamente en sus niveles de ingresos.

Paso 2: Preparación de los Datos

Una vez identificadas las características necesarias, el siguiente paso fue preparar los datos para su uso en el modelo predictivo. Esto incluyó la transformación de las variables categóricas en numéricas, esto para que mi modelo pueda interpretar los valores de mis features de una manera más clara y consistente.

etl > dictionaries.py > ...

```
workclass_dict = {
    'Private': 1, 'Self-emp-not-inc': 2, 'Self-emp-inc': 3, 'Federal-gov': 4,
    'Local-gov': 5, 'State-gov': 6, 'Without-pay': 7, 'Never-worked': 8
}
marital_status_dict = {
    'Married-civ-spouse': 1, 'Divorced': 2, 'Never-married': 3, 'Separated': 4,
    'Widowed': 5, 'Married-spouse-absent': 6, 'Married-AF-spouse': 7
}
occupation_dict = {
    'Tech-support': 1, 'Craft-repair': 2, 'Other-service': 3, 'Sales': 4,
    'Exec-managerial': 5, 'Prof-specialty': 6, 'Handlers-cleaners': 7,
    'Machine-op-inspct': 8, 'Adm-clerical': 9, 'Farming-fishing': 10,
    'Transport-moving': 11, 'Priv-house serv': 12, 'Protective-serv': 13,
    'Armed-Forces': 14
}
```

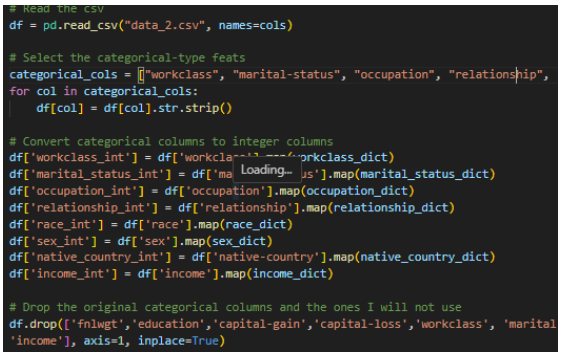
Figura 1. Transformación de variables categóricas en numéricas

Paso 3: Limpieza y Extracción de Datos

El proceso de limpieza de datos consistió en eliminar filas con valores faltantes (nulos) que podían comprometer la calidad del análisis. A pesar de la pérdida de un pequeño porcentaje de los datos (alrededor de un 7% de las instancias del dataset), esta decisión fue clave para asegurar la precisión y la fiabilidad del modelo. Después de la limpieza, los datos

fueron almacenados en un nuevo conjunto, listo para ser utilizado en los siguientes pasos del análisis.

Este proceso de análisis es crucial para garantizar que el modelo predictivo se base en datos sólidos y relevantes, lo que maximiza su capacidad para identificar patrones y tendencias que pueden influir en los ingresos de las personas.




```
# Read the csv
df = pd.read_csv("data_2.csv", names=cols)

# Select the categorical-type feats
categorical_cols = ["workclass", "marital-status", "occupation", "relationship",
for col in categorical_cols:
    df[col] = df[col].str.strip()

# Convert categorical columns to integer columns
df['workclass_int'] = df['workcla...orkclass_dict)
df['marital_status_int'] = df['ma...us'].map(marital_status_dict)
df['occupation_int'] = df['occupation'].map(occupation_dict)
df['relationship_int'] = df['relationship'].map(relationship_dict)
df['race_int'] = df['race'].map(race_dict)
df['sex_int'] = df['sex'].map(sex_dict)
df['native_country_int'] = df['native-country'].map(native_country_dict)
df['income_int'] = df['income'].map(income_dict)

# Drop the original categorical columns and the ones I will not use
df.drop(['fnlwgt','education','capital-gain','capital-loss','workclass', 'marital
'income'], axis=1, inplace=True)
```

Figura 2. Extracción de datos del set de datos y elección de columnas a utilizar en el modelo, así como la transformación de las mismas.

III. Set de Datos

El conjunto de datos utilizado en este análisis, conocido como "Census Income" o "Adult," es utilizado en estudios de clasificación dentro del campo de las ciencias sociales. Este dataset multivariado contiene 32,560 instancias y 14 características originales, aunque en el modelo final se utilizaron 30,019 instancias y una selección específica de features.

Características del Dataset
Área: Ciencias Sociales
Tareas Asociadas: Clasificación
Tipos de Features: Categóricas y Enteras
Características Seleccionadas para el Modelo Final:

- age (Edad)
- workclass (Clase de Trabajo)
- education-num (Nivel Educativo)
- marital-status (Estado Civil)
- occupation (Ocupación)
- relationship (Relación Familiar)
- race (Raza)
- sex (Sexo)
- hours-per-week (Horas Trabajadas por Semana)
- native-country (País de Origen)
- income (Ingreso)

Selección de Features
Las columnas seleccionadas para el modelo final fueron elegidas debido a su relevancia en la predicción de ingresos. Estas variables representan una combinación de factores demográficos, educativos, laborales y sociales que tienen un impacto directo en los niveles de ingresos de los individuos.

Es importante recalcar que las columnas 'fnlwgt', 'capital-gain', y 'capital-loss' no fueron consideradas debido a la falta de descripción y contexto suficiente para comprender sus características, valores y cómo éstas tenían relación directa con el objetivo del dataset. La columna 'education' fue omitida ya que sus valores eran redundantes al contar con la feature education-num.

Este dataset es un excelente recurso para analizar cómo diferentes aspectos de la vida de una persona pueden influir en su situación económica, proporcionando un marco sólido para la creación de modelos predictivos en el ámbito de las ciencias sociales.

IV. Propuesta de Modelo

En esta sección, se presenta el modelo de regresión logística implementado para predecir si una persona tiene ingresos superiores a $50,000 anuales.

Regresión logística

Para dar un poco de contexto, la regresión logística es una técnica de modelado estadístico utilizada para predecir la probabilidad de un evento binario, es decir, un resultado que puede tener dos posibles valores (por ejemplo, sí/no, 0/1, verdadero/falso).

El modelo se basa en varios conceptos matemáticos y estadísticos clave, que se describen a continuación:

Función de Hipótesis: Sigmoide

La función sigmoide se utiliza como la función de hipótesis en nuestro modelo. Matemáticamente, se define como:

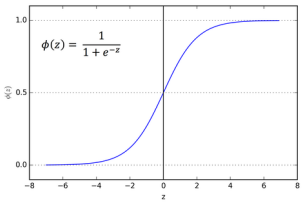


Figura 3. Función sigmoide

Esta función convierte cualquier valor real en un rango entre 0 y 1, lo que la hace ideal para modelos de clasificación binaria. En nuestro caso, $\sigma(z)$ representa la probabilidad de que una instancia pertenezca a la clase con ingreso >$50,000.

La sigmoide facilita la interpretación de las predicciones como probabilidades y es ideal para problemas de clasificación binaria.

Función de Pérdida: Entropía Cruzada

La entropía cruzada es la medida utilizada para calcular la pérdida o error del modelo. Esta métrica evalúa qué tan bien se están ajustando las predicciones a las etiquetas verdaderas. La fórmula general para la entropía cruzada es:

$$L(y, \hat{y}) = -\left(y \log(\hat{y}) + (1-y) \log(1-\hat{y})\right)$$

Figura 4. Función de pérdida entropía cruzada

En este modelo, se utiliza una versión ponderada de la entropía cruzada para dar más peso a las clases desbalanceadas (es decir, a las instancias con ingresos superiores a $50,000, que son menos comunes).

La entropía cruzada ponderada nos ayuda a tener una visualización sobre la capacidad del modelo para manejar conjuntos de datos desequilibrados, asegurando que las predicciones sean precisas y justas para ambas clases.

Optimización: Descenso de Gradiente

El descenso de gradiente es la técnica utilizada para optimizar los parámetros del modelo. Este método ajusta los parámetros para minimizar la función de pérdida. En cada iteración, los parámetros se actualizan en la dirección opuesta al gradiente de la función de pérdida con respecto a esos parámetros. La tasa de aprendizaje (learning rate, lr) controla el tamaño del paso que damos en cada iteración:

$$\Theta_j := \Theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} [(h_\Theta(x_i) - y) x_i]$$

Figura 5. Función de optimización de parámetros de descenso de gradiente.

Este proceso se repite hasta que la función de pérdida converge a un valor mínimo.

El descenso de gradiente ofrece un método eficiente para minimizar la función de pérdida, garantizando que el modelo aprenda de manera efectiva.

Normalización de Datos

En la construcción de modelos de machine learning, la normalización de los datos es un paso crucial, especialmente cuando los features tienen diferentes escalas. Al principio, no implementé la normalización en mi algoritmo, ya que no le veía mucho valor. Sin embargo, al analizar los resultados, noté que el algoritmo favorecía considerablemente las instancias con un target de 0 en comparación con las de 1. La mayoría de las predicciones resultaban en 0, lo que indicaba un sesgo en el modelo.

Al investigar más a fondo, identifiqué que este problema se debía a las columnas 'native-country' y 'hours-per-week', que contenían valores significativamente más altos

que otras columnas. Este desequilibrio en las escalas de los datos hizo que el algoritmo asignara más peso a las instancias con valores altos en estas columnas, lo que generaba predicciones sesgadas.

Para corregir este problema, implementé la función de normalización. La normalización transforma los datos para que todas las features tengan la misma escala, eliminando el sesgo inducido por las diferencias en magnitudes entre las columnas. La fórmula que utilicé para normalizar los datos es:

$$X_{\text{normalizado}} = \frac{X - \text{media}(X)}{\text{desviación estándar}(X)}$$

Figura 6. Función de normalización

Donde X representa los datos originales, media(X) es la media de cada feature y desviación estándar(X) es la desviación estándar de cada feature.

Esta transformación garantiza que cada feature tenga una media de 0 y una desviación estándar de 1, permitiendo que el algoritmo considere todas las features de manera equitativa. Después de aplicar esta normalización, el modelo dejó de favorecer instancias específicas y comenzó a hacer predicciones más equilibradas y precisas.

Primer Intento: Regresión Logística Deprecada

En el repositorio de GitHub, tengo un archivo llamado "logistic_reg_deprecated.py," que contiene mi primer intento de implementar la regresión logística. Este intento no funcionó del todo bien; aunque los parámetros se actualizaban correctamente, el proceso era extremadamente lento. El modelo tardó una hora en reducir el error a apenas 0.5.

El problema principal fue la implementación de batches. Un batch es un subconjunto del conjunto de datos completo utilizado para actualizar los parámetros en cada iteración. Sin embargo, este enfoque llevó a un overfitting, donde el error disminuía y luego aumentaba repetidamente. Esto indicaba que el modelo había aprendido patrones específicos de ciertos batches, lo que lo atrapaba en un bucle sin mejorar realmente su capacidad de generalización.

Algoritmo Optimizado

El nuevo algoritmo, "log_reg.py," incluye mejoras significativas sobre el primer intento. El uso de normalización y la eliminación de batches problemáticos han permitido que el modelo sea más eficiente y generalice mejor los patrones en el conjunto de datos. Ahora, el error converge de manera más consistente, lo que indica un modelo más robusto y confiable.

V. Train y Test

En machine learning, separar un dataset en conjuntos de entrenamiento (train) y prueba (test) es una práctica esencial para evaluar la capacidad de un modelo para generalizar a nuevos datos. Al entrenar un modelo únicamente en un subconjunto de los datos disponibles y luego probarlo en un conjunto separado, podemos estimar cómo funcionará el modelo en datos no vistos.

En mi algoritmo, implementé esta separación de la siguiente manera:

1. **Preparación de los Datos**:

   Primero, extraigo las features (x) y el target (y) del dataset limpio. El target, ‘income’, es la variable que queremos predecir.

2. **Shuffle de los Datos**:

   Antes de dividir el dataset en train y test, hago un shuffle (mezcla aleatoria) de los datos. Esto es crucial porque garantiza que la distribución de las instancias en los conjuntos de entrenamiento y prueba sea

representativa de la distribución total del dataset. Sin el shuffle, podríamos introducir un sesgo si, por ejemplo, las primeras filas del dataset estuvieran ordenadas de alguna manera.

3. **División en Train y Test**:

   Luego, divido los datos manualmente utilizando un 'train_ratio' del 80%, lo que significa que el 80% de los datos se utilizan para entrenar el modelo y el 20% restante para probarlo. Esta es una proporción comúnmente utilizada y proporciona un buen equilibrio entre la cantidad de datos para entrenar y evaluar el modelo.

4. **Normalización**:

   Después de dividir los datos en conjuntos de entrenamiento y prueba, aplico la normalización a ambos conjuntos. La normalización se realiza después de la división para evitar cualquier fuga de información entre el train y test, lo que podría conducir a un modelo con un rendimiento engañoso.

5. **Calibración del Modelo**:

   Calculé los pesos de las clases para manejar el desequilibrio en las clases, dado que los modelos pueden sesgarse hacia la clase mayoritaria, que en este caso, tengo más instancias con un target de 0 que de 1 (en mis cálculos es un ratio de 3:1 favoreciendo a las instancias con target de 0). Luego, inicialicé los parámetros que se optimizarían durante el entrenamiento del modelo.

6. **Entrenamiento del Modelo**:

   Utilicé el algoritmo de Gradient Descent para ajustar los parámetros del modelo durante un número fijo de iteraciones (epochs). A medida que el modelo se entrena, monitorizo el error usando la función de pérdida de cross-entropy ponderada.

7. **Predicciones**:

   Después del entrenamiento, realizo predicciones tanto en el conjunto de entrenamiento (train) como en el de prueba (test). Establecí un límite de 0.38 para clasificar las predicciones como 0 o 1, esto debido a que la mayoría de las predicciones estarán abajo de 0.5 porque hay el triple de instancias con target de 0 que de 1.

La división en conjuntos de entrenamiento y prueba, combinada con una evaluación cuidadosa de los resultados en ambos, asegura que el modelo tenga la capacidad de generalizar bien y no solo memorizar patrones específicos de los datos de entrenamiento (algo que me sucedió en mi primer implementación). Esto es fundamental para desarrollar modelos de machine learning robustos y confiables.

VI. Resultados

En esta sección, introduzco algunos elementos visuales que ayudan a interpretar los resultados de mi algoritmo de regresión logística.

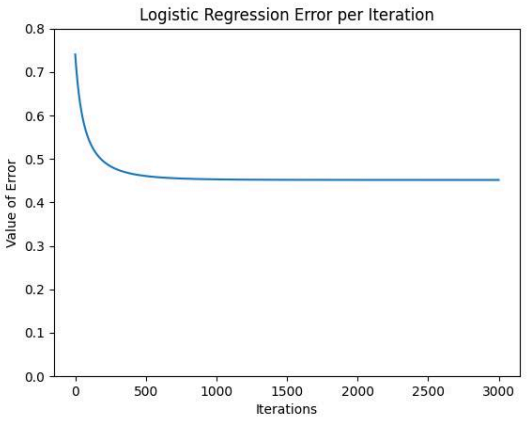


Figura 7. Error del modelo por iteración

La gráfica de error por iteración muestra cómo la pérdida de entropía cruzada ponderada

disminuye a lo largo de las iteraciones durante el entrenamiento del modelo de regresión logística.

La gráfica indica que el modelo ha alcanzado la convergencia, lo que significa que ha encontrado un conjunto de parámetros que minimizan la pérdida de manera efectiva.

A pesar de la estabilización, es importante seguir evaluando el rendimiento del modelo en los datos de prueba para asegurarse de que no esté sobreajustado, en otras palabras, que no esté haciendo un overfitting, que es justamente lo que se muestra en las siguientes gráficas.

Figura 8. Matriz de confusión para datos de entrenamiento

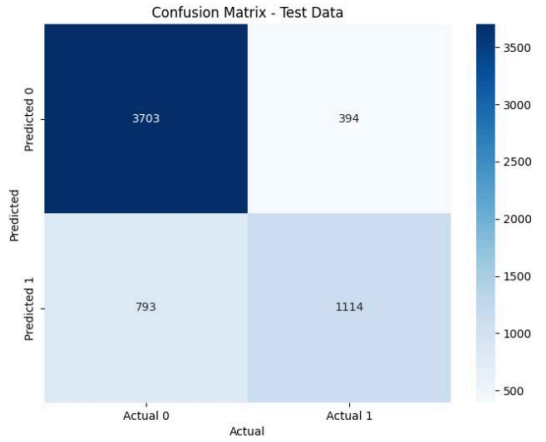


Figura 9. Matriz de confusión para datos de prueba

Las dos matrices de confusión anteriores representan la cantidad de predicciones correctas e incorrectas en mis datos de entrenamiento y de prueba.

Comparando las matrices de confusión de los datos de entrenamiento y prueba, podemos observar que el rendimiento del modelo es consistente en ambos conjuntos de datos. Esto sugiere que el modelo está generalizando bien y no está haciendo overfitting.

A continuación, daré mi interpretación a cada una de las posibles categorías que a mi algoritmo se le pueden asignar:

1. **Overfitting:** No hay una gran discrepancia entre las métricas de entrenamiento y prueba, lo que indica que el modelo no está sobreajustado.

2. **Underfitting:** El modelo está capturando bien las relaciones en los datos, ya que las métricas son razonablemente buenas en ambos conjuntos de datos. Esto fue gracias a los pesos que le asigné a ambas clases en el entrenamiento del modelo y al umbral que definí para determinar si una predicción es clase 0 o 1, aunque tengo que admitir que todavía hay espacio de mejora, en especial para la clase 1.

3. **Fitting**: El rendimiento en los datos de prueba es similar al de los datos de entrenamiento, lo que sugiere que el modelo está ajustando bien y generalizando adecuadamente.

VII. Conclusiones

Para finalizar, he llegado a la conclusión que el modelo está ajustando bien (fitting) debido a que las métricas de rendimiento en los datos de prueba son similares a las de los datos de entrenamiento, lo que indica que el modelo ha aprendido las relaciones subyacentes en los datos sin sobre ajustarse ni sub ajustarse. La gráfica de error por iteración también respalda esta conclusión, mostrando una disminución

rápida y estabilización del error, lo que sugiere que el modelo ha alcanzado la convergencia de manera efectiva.

Además, la consistencia en el rendimiento del modelo entre los conjuntos de datos de entrenamiento y prueba refuerza la idea de que los parámetros de mi modelo han sido sintonizados adecuadamente el cómo se relacionan los conjuntos de datos. Esta estabilidad dice que el modelo no solo ha aprendido correctamente los patrones presentes en el conjunto de entrenamiento, sino que también tiene la capacidad de generalizar esos patrones a datos no vistos previamente, algo que es sumamente valioso ya que sugiere que el modelo puede hacer predicciones acertadas con información nueva.

Sin embargo, es importante considerar que, aunque el modelo está mostrando un buen rendimiento, siempre existe la posibilidad de mejorar. Por ejemplo, ajustar más finamente los pesos asignados a las clases o explorar diferentes umbrales de clasificación podría aumentar la precisión, especialmente en la clase minoritaria, que en este caso es la clase 1. Asimismo, probar diferentes técnicas de regularización o realizar un análisis más exhaustivo de la matriz de confusión podría ayudar a identificar áreas de mejora para asegurar un rendimiento aún más robusto y equilibrado.

En resumen, los resultados obtenidos hasta el momento sugieren que el modelo está bien calibrado y tengo que decir que es un trabajo del que me siento muy orgulloso, aún estando consciente de sus áreas de oportunidad.

VIII. Bibliografía

¿Qué es la regresión logística? - Explicación del modelo de regresión logística - AWS. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/logistic-regression/

UCI Machine Learning Repository. (s. f.). https://archive.ics.uci.edu/dataset/2/adult